* UML Diagram :

- Unified Modelling Language.

- It is a standard language for writing s/w blueprint.

- UML is a language for visualizing, speciting, constructing & documenting artifacts of s/w system.

* Visualising :

- An explicit model facilitates communications.

- Each symbol has well defined [illegible] sementics behind it.

* Specifing :

- UML addresses specification of all important analysis, design & implementation decision.

* Constructing :

1) forward engineering is a process of generating code from model into programming language.

2) Backword engineering [reverse] is a process of reconstructing model from implementations

3) roundtrip engineering is going both the way.

* documenting :

- Artifacts include,
  - Deliver-ables such as requirement document, functions specification & Test plan
  - Materials data critical in controlling, measuring & communicating about a system during development & After deployment

* Importance of UML :

1) It provide structure for problem solving.

2) It reduces time to market for business problem solution.

3) It decreases development ~~[illegible]~~. cost.

4) Keep manage risk of mistakes.

* UseCase Diagram :

- It describe what a system does from stand point of external observer.

- It emphasize on what a system does rather than how it does

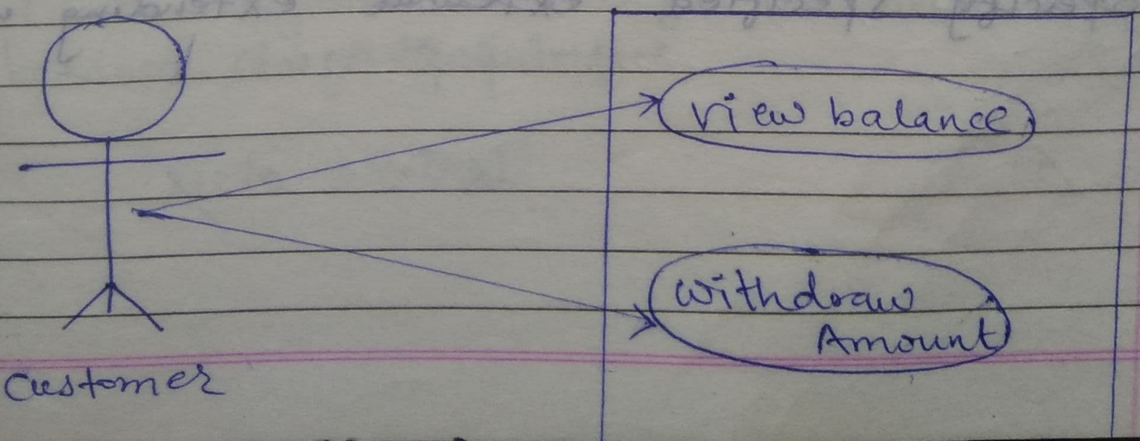
- A usecase diagram is a collection of actors, usecases & their communications.

- Usecase is a sequence of action including variante that a system perform to obtain observable result of value to the customer.

- Actor is a logical set of roles that human and/or Non-human user of use cases play while interacting with those usecases

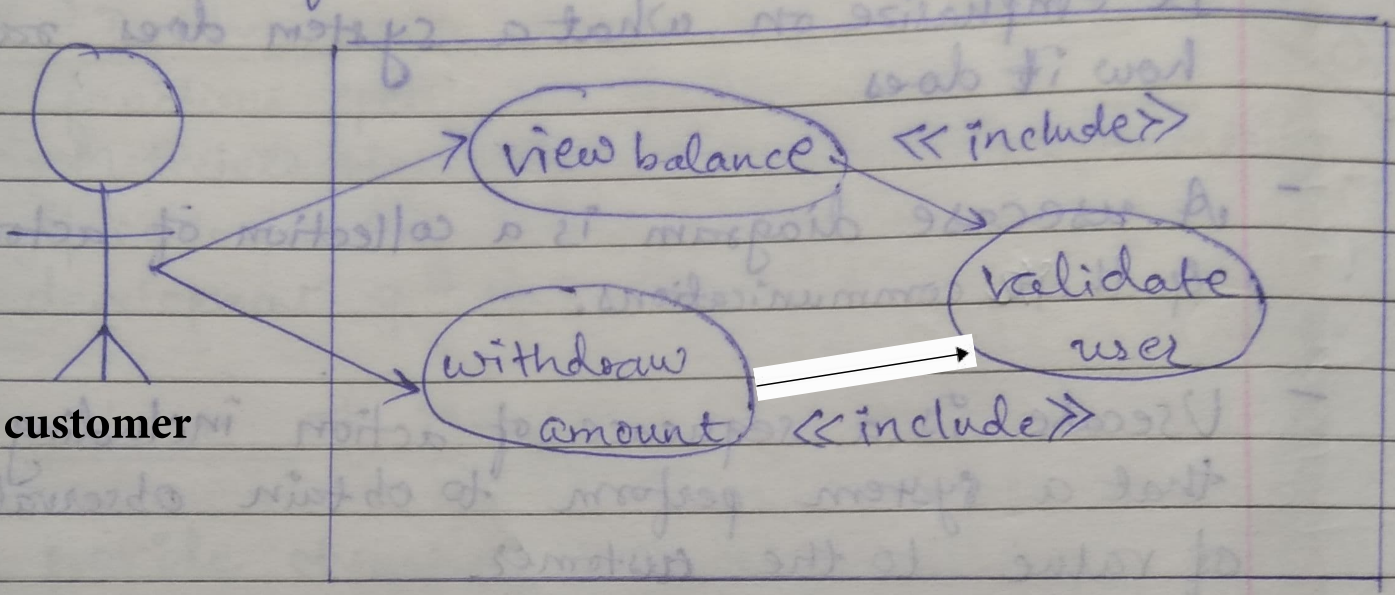
- An actor is usually drawn as named stick figure

- ~~fog~~ eg. In use case of ATM system, there is a customer who performed viewbalance, operation & withdrawal amount.



* include:

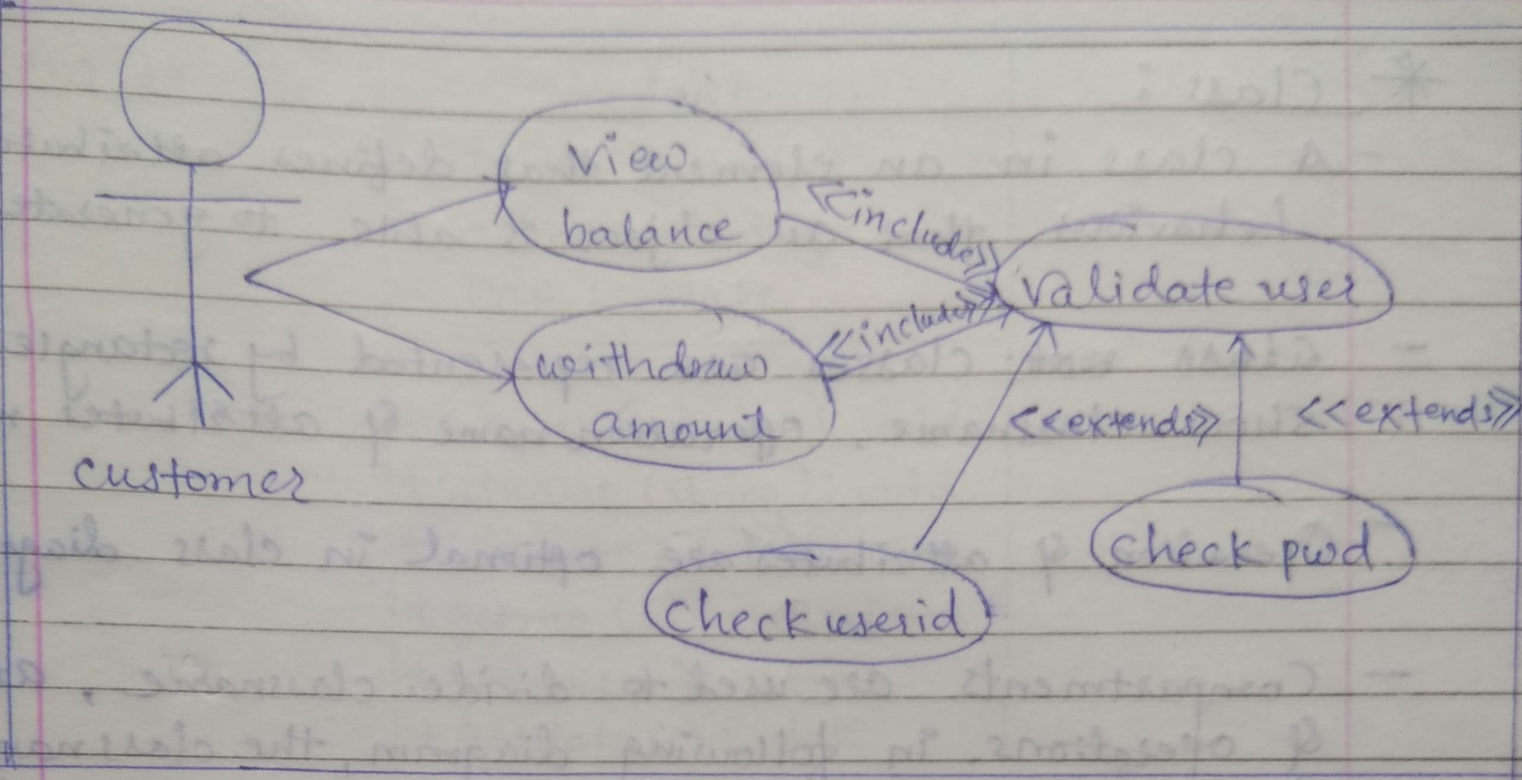
- we can use <<include>> stereotype to indicate that based used case "include" ~~[illegible]~~ contents or behaviours of common use case.



- Usecases may be included by one or more usecases, helping to reduce level of duplication of functionality by factoring out common behaviour into usecases that are reused many times.

* Extends:

- we can use <<extends>> sterotype to indiate that base use case many include behaviour ~~specify~~ specified ~~external~~ extending usecases.



* Class Diagram :

- Class diagram represent building block of any Object Oriented System.

- Class diagram depicts static view of model.

- Class diagram describe what attributes & behaviour it has rather than detailing the methods for acheiving operations.

- Class diagram are useful in illustrating relationship between classes & Interface.

* Class :

- A class in an element that defines attributes & behaviour that an object is able to generate.

- ~~Class~~ ~~root~~: classes are represented by rectangles with show classname, operation name & attributes name.

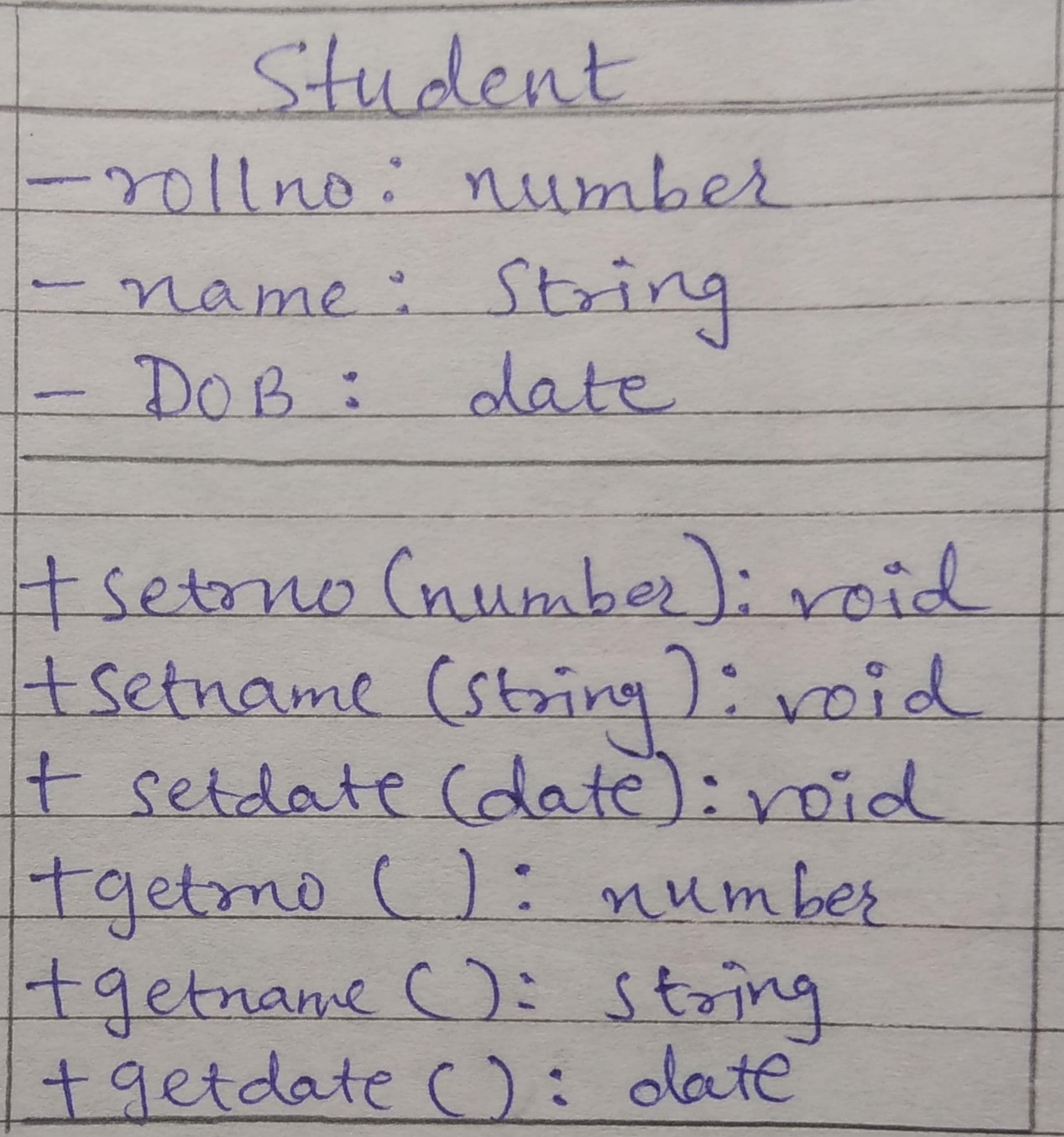
- Operation & attributes are optional in class diagram.

- Compartments are used to divide classname, attributes & operations. in following diagram, the classname in top most compartment, next most compartment details attribute & final compartment show the operation.

- The notation that preeed attribute & operation name indicates visibility of element

	Symbol	Visibility
1)	+	public
2)	−	private
3)	#	protected
4)	~	package
5)	/	Derived class

for eg.



Student

-rollno: number

- name: String

- DOB: date

+setrno (number): void

+setname (string): void

+ setdate (date): void

+getrno ( ): number

+getname (): string

+getdate (): date